5

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/697,865	10/30/2003	Michael Norman Day	AUS920030693US1	9752

45327 7590 12/22/2006
IBM CORPORATION (CS)
C/O CARR LLP
670 FOUNDERS SQUARE
900 JACKSON STREET
DALLAS, TX 75202

EXAMINER	
SCHELL, JOSEPH O	
ART UNIT	PAPER NUMBER
2114	

SHORTENED STATUTORY PERIOD OF RESPONSE	MAIL DATE	DELIVERY MODE
3 MONTHS	12/22/2006	PAPER

Please find below and/or attached an Office communication concerning this application or proceeding.

If NO period for reply is specified above, the maximum statutory period will apply and will expire 6 MONTHS from the mailing date of this communication.

	Application No.	Applicant(s)	
Office Action Summary	10/697,865	DAY ET AL.	
	Examiner	**Art Unit**	
	Joseph Schell	2114	

-- *The MAILING DATE of this communication appears on the cover sheet with the correspondence address* --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on 30 October 2003.

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) 1-10 is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) 1-10 is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☒ The drawing(s) filed on 30 October 2003 is/are: a)☒ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☐ Information Disclosure Statement(s) (PTO/SB/08) Paper No(s)/Mail Date _____.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____ .
5) ☐ Notice of Informal Patent Application
6) ☐ Other: _____.

Detailed Action

Claims 1-10 have been examined.

Claims 1-10 have been rejected.

Claim Objections

1. Line 1 of Claims 3, 4 and 8 start with "The method of debugging..." These claims are independent and should begin with "A method.."

2. In Claim 4, line 25 (on the page), the term "the debugging process" is used. This term lacks antecedent basis.

Claim Rejections - 35 USC § 112

The following is a quotation of the second paragraph of 35 U.S.C. 112:

The specification shall conclude with one or more claims particularly pointing out and distinctly claiming the subject matter which the applicant regards as his invention.

3. Claims 1-2, 5 and 7-10 are rejected under 35 U.S.C. 112 ~~first~~ 2nd paragraph for being indefinite.

4. Within Claim 1, lines 2-3 of the claim state "a first processor means for providing said processor with access..." In this case the limitation of "said processor" lacks antecedent basis. Additionally, "a first processor means for providing said first processor means with access" should be shortened to "a first processor means for

accessing" since a processor providing itself with access is just an unclear, irregular way of saying the same thing.

5. Within Claim 5, lines 14-15 of the second page of the claim state "parameters of said specified program in said pool of memory." However, what is inserted into the pool of memory is register states of the supplemental processor executing the specified program. It is unclear what is intended by this use of "parameters." The examiner is assuming that it is meant to mean whatever program-related data is stored in the status registers of the processor executing said program at the time of its interruption.

6. Claims 7-10 contain the same "parameters" antecedent basis and consistency issues as Claim 5.

Claim Rejections - 35 USC § 103

The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

7. Claims 1-4 and 8-10 are rejected under 35 U.S.C. 103(a) as being unpatentable over Quach (US Patent 6,625,749) in view of Hashemi (US Patent 5,491,787).

8. As per claim 1, Quach ('749) discloses a method for use in a computer system having a main memory and a first processor means (as shown in Figure 1, there are two Execution Cores) for providing said processor with access to the register state of a supplemental processor (column 10 lines 30-40, the control and status registers are checked. This checking is not explicitly performed by the other processor, but this limitation is addressed using Hashemi ('787), below), said supplemental processor otherwise inaccessible by said first processor, comprising:

loading a program into said supplemental processor (as shown in Figure 3, the program is running on both processing cores while a discrepancy is not detected (step 31), to run a program it must be loaded);

executing said program in said supplementary processor to generate said register states of said supplementary processor (Figure 3 step 31);

storing said register states in said main memory (Figure 4 step 440); and

accessing said register state by said first processor (Figure 4 step 490).

In the system disclosed by Quach ('749), the saved states of both the first and second processors are parity-checked for valid complete or partial states (column 8 lines 33-38). Quach ('749) does not expressly disclose which processor performs the checking of states, and thus does not strictly anticipate that the state of a first processor is checked by a second processor. Each state might be parity-checked by its own processor.

Hashemi ('787) teaches a dual-processor system wherein the two processors alternate between shadowing and main operation (see abstract). Within this system, the shadow processor checks the execution of the other processor (column 3 lines 48-51).

At the time of invention it would have been obvious to a person of ordinary skill in the art to modify the processor-state checking system disclosed by Quach ('749) such that the two processors don't each parity check their own saved state. This modification would have been obvious because if a checking processor is faulty it may not detect faults the processor it is checking (Hashemi ('787) column 1 line 66 through column 2 line 4) and thus the debugging processor also needs to be checked for correct operation (Hashemi ('787) column 3 lines 19-21).

9. As per claim 2, Quach ('749) in view of Hashemi ('787) discloses the method of claim 1 further including a program debugger operating on said main processor and wherein said method further includes:

accessing said register state in said main memory by said debugger to debug said program (As shown in Quach ('749) figure 4, elements 440-460, register states are saved to memory and are then checked for parity errors. In view of Hashemi ('786), it would have been obvious that the parity checking of the saved register states be done by the other processor. This would have been obvious because if the processors do not check each other than a faulty processor may no detect faults in itself (Hashemi ('787) column 1 line 66 through column 2 line 4).

10. As per claim 3, Quach ('749) discloses the method of debugging a specified program that has been operationally interrupted (column 9 lines 1-4), nominally operating on a supplemental processor (see Figure 1, there are two processing cores), the register states of which cannot be directly accessed by a main processor operating a debugging program comprising:

activating a secondary program in said supplemental processor to transmit register states of said supplemental processor, at the time said specified program is operationally interrupted, to memory (column 9 lines 5-21, the ERR routine stops the processor from further executing it's program and as shown in Figure 4 step 440, the register states are saved to memory);

modifying parameters of said specified program in a memory pool accessible by said main processor (column 10 lines 22-27); and

restoring operation of said specified program in said supplemental processor containing alterations generated in the debugging process (Figure 3 steps 380 and 390).

In the system disclosed by Quach ('749), the saved states of both the first and second processors are parity-checked for valid complete or partial states (column 8 lines 33-38). Quach ('749) does not expressly disclose the system wherein the register states are transmitted to the main processor and a debugging program on the main processor is used to debug the states.

Hashemi ('787) teaches a dual-processor system wherein the two processors alternate between shadowing and main operation (see abstract). Within this system, the shadow processor checks the execution of the other processor (column 3 lines 48-51).

At the time of invention it would have been obvious to a person of ordinary skill in the art to modify the processor-state checking system disclosed by Quach ('749) such that the two processors don't each parity check their own saved state. This modification would have been obvious because if a checking processor is faulty it may not detect faults the processor it is checking (Hashemi ('787) column 1 line 66 through column 2 line 4) and thus the debugging processor also needs to be checked for correct operation (Hashemi ('787) column 3 lines 19-21).

11. As per claim 4, Quach ('749) discloses the method of debugging a specified program intended to operate on a supplemental processor (see Figure 1, either core may be considered the supplemental processor), said supplemental processor having limited memory and flexibility (the processors anticipated by Quach ('749) do not have unlimited memory or flexibility), the register states of which cannot be directly accessed by a read command from another processor comprising:

reserving a pool of memory accessible to a debugging program processor (column 10 lines 21-24);

running a specified program, to be debugged, in said supplemental processor until instructions in said specified program cause operation of said specified program to cease (column 8 line 64 through column 9 line 2, the processor is executing a program in redundant mode until an error is detected, whereupon the processor jumps to an ERR);

activating a secondary program in said supplemental processor to transmit register states of said supplemental processor, at the time said specified program is operationally interrupted, to memory (column 9 lines 5-15);

modifying parameters of said specified program in a memory pool accessible by said debugging program processor (column 10 lines 22-27); and

restoring operation of said specified program in said supplemental processor with program alterations modified in the debugging process (Figure 3 steps 380 and 390).

In the system disclosed by Quach ('749), the saved states of both the first and second processors are parity-checked for valid complete or partial states (column 8 lines 33-38). Quach ('749) does not expressly disclose the system wherein the register states are transmitted to the main processor and a debugging program on the main processor is used to debug the states.

Hashemi ('787) teaches a dual-processor system wherein the two processors alternate between shadowing and main operation (see abstract). Within this system, the shadow processor checks the execution of the other processor (column 3 lines 48-51).

At the time of invention it would have been obvious to a person of ordinary skill in the art to modify the processor-state checking system disclosed by Quach ('749) such that the two processors don't each parity check their own saved state. This modification would have been obvious because if a checking processor is faulty it may not detect faults the processor it is checking (Hashemi ('787) column 1 line 66 through column 2 line 4) and thus the debugging processor also needs to be checked for correct operation (Hashemi ('787) column 3 lines 19-21).

12. As per claim 8, Quach ('749) discloses the method of debugging a first processor unit employing a second processor operating a debugging program comprising:

activating a secondary program in first processor to transmit register states of said first processor, subsequent to a time a specified program to be debugged is operationally interrupted, to memory (as shown in Figure 5b, the processor state is saved to memory where it is modified by the other processor with the correct register state, see also column 9 lines 5-15);

modifying parameters of said specified program in a memory pool accessible by said second processor through the use of said debugging program in said second processor (column 10 lines 22-27); and

restoring operation of said specified program in said first processor with alterations as created in the debugging process (Figure 3 steps 380 and 390).

In the system disclosed by Quach ('749), the saved states of both the first and second processors are parity-checked for valid complete or partial states (column 8 lines 33-38). Quach ('749) does not expressly disclose the system wherein the register states are transmitted to the main processor and a debugging program on the main processor is used to debug the states.

In the system disclosed by Quach ('749), the saved states of both the first and second processors are parity-checked for valid complete or partial states (column 8 lines 33-38). Quach ('749) does not expressly disclose the system wherein the register states are transmitted to the main processor and a debugging program on the main processor is used to debug the states.

Hashemi ('787) teaches a dual-processor system wherein the two processors alternate between shadowing and main operation (see abstract). Within this system, the shadow processor checks the execution of the other processor (column 3 lines 48-51).

At the time of invention it would have been obvious to a person of ordinary skill in the art to modify the processor-state checking system disclosed by Quach ('749) such that the two processors don't each parity check their own saved state. This modification would have been obvious because if a checking processor is faulty it may not detect faults the processor it is checking (Hashemi ('787) column 1 line 66 through column 2 line 4) and thus the debugging processor also needs to be checked for correct operation (Hashemi ('787) column 3 lines 19-21).

13. As per claim 9, Quach ('749) discloses a computer program product for debugging a first processor employing a supplemental processor (Figure 1, the two execution cores), the computer program product having a medium with a debugging computer program embodied thereon, the debugging computer program comprising:

computer code for transmitting and activating a secondary program in a first processor transmitting register states of said first processor, subsequent to the operational halting of said program requiring debugging, to said second processor (column 8 line 64 through column 9 line 2, the processor is executing a program in redundant mode until an error is detected, whereupon the processor jumps to an ERR and column 9 lines 5-15);

computer code for modifying parameters of said specified program in a memory pool accessible by said second processor through the use of said debugging program in said second processor (column 10 lines 22-27); and

computer code for restoring operation of said specified program in said first processor with alterations generated by the debugging process (Figure 3 steps 380 and 390).

In the system disclosed by Quach ('749), the saved states of both the first and second processors are parity-checked for valid complete or partial states (column 8 lines 33-

38). Quach ('749) does not expressly disclose the system wherein the register states are transmitted to the main processor and a debugging program on the main processor is used to debug the states.

Hashemi ('787) teaches a dual-processor system wherein the two processors alternate between shadowing and main operation (see abstract). Within this system, the shadow processor checks the execution of the other processor (column 3 lines 48-51).

At the time of invention it would have been obvious to a person of ordinary skill in the art to modify the processor-state checking system disclosed by Quach ('749) such that the two processors don't each parity check their own saved state. This modification would have been obvious because if a checking processor is faulty it may not detect faults the processor it is checking (Hashemi ('787) column 1 line 66 through column 2 line 4) and thus the debugging processor also needs to be checked for correct operation (Hashemi ('787) column 3 lines 19-21).

14. As per claim 10, Quach ('749) discloses a computer program product for authenticating code in a computer system, the computer program product having a medium with a computer program embodied thereon, the computer program comprising:

computer code for transmitting and activating a secondary program in a first processor transmitting register states of said first processor, subsequent to the operational halting of said program requiring debugging, to said second processor (column 9 lines 5-15);

computer code for modifying parameters of said specified program in a memory pool accessible by said second processor through the use of said debugging program in said second processor (column 10 lines 22-27); and

computer code for restoring operation of said specified program in said first processor with alterations generated by the debugging process (Figure 3 steps 380 and 390).

In the system disclosed by Quach ('749), the saved states of both the first and second processors are parity-checked for valid complete or partial states (column 8 lines 33-38). Quach ('749) does not expressly disclose the system wherein the register states are transmitted to the main processor and a debugging program on the main processor is used to debug the states.

Hashemi ('787) teaches a dual-processor system wherein the two processors alternate between shadowing and main operation (see abstract). Within this system, the shadow processor checks the execution of the other processor (column 3 lines 48-51).

At the time of invention it would have been obvious to a person of ordinary skill in the art to modify the processor-state checking system disclosed by Quach ('749) such that the two processors don't each parity check their own saved state. This modification would have been obvious because if a checking processor is faulty it may not detect faults the processor it is checking (Hashemi ('787) column 1 line 66 through column 2 line 4) and thus the debugging processor also needs to be checked for correct operation (Hashemi ('787) column 3 lines 19-21).

15. Claims 5 and 7 are rejected under 35 U.S.C. 103(a) as being unpatentable over Bebout (US Patent 7,124,404) in view of Wikipedia's RAM and Moreton (US Patent Application Publication 2004/0207630).

16. As per claim 5, Bebout ('404) discloses Debugging apparatus comprising:

a debugging processor (as shown in Figure 3, element 310);

a supplemental processor (as shown in Figure 3, element 340);

a specified program to be debugged installed for operation in said supplemental processor (column 2 lines 9-13);

a communication path between and interconnecting said supplemental processor and said debugging processor (as shown in Figure 3, elements 322 and 323); and

a debugging program operable to,

activate a secondary program in said supplemental processor to transmit register states of said supplemental processor, at the time said specified program is operationally interrupted, to said debugging processor (column 6 lines 39-44);

modify parameters of said specified program through the use of said debugging program in said debugging processor (column 6 lines 57-60), and restore operation of said specified program in said supplemental processor with alterations modified in the debugging process (column 4 lines 66, the run command is available to resume operation).

Bebout ('404) discloses that the debugger can modify the state of its respective CPU (column 5 lines 10-14), that the debugger allows the user to diagnose the status of the system (column 6 lines 39-40) and that the debugger has an updateable state that can reflect the current state of a CPU (column 7 lines 13-15). However, Bebout ('404) does not expressly disclose the debugging apparatus wherein the debug processor has an available pool of memory that is used for debug operations.

Wikipedia's RAM teaches general information about random access memory.

At the time of invention it would have been obvious to a person of ordinary skill in the art to modify the debug system disclosed by Bebout ('404) such that the debugger stores CPU state information to a memory. This modification would have been obvious because computers use RAM to hold program code and data during execution (Wikipedia's RAM, first sentence under Overview).

Bebout ('404) in view of Wikipedia's RAM does not disclose the use of reserving memory.

Moreton ('630) teaches a method of arranging data within memory (see abstract).

At the time of invention it would have been obvious to a person of ordinary skill in the art to modify the debug system disclosed by Bebout ('404) in view of Wikipedia's RAM such that a set of addresses are reserved for debug operations. This modification would have been obvious because when a memory location is accessed its neighboring locations are accessed simultaneously (Moreton ('630) paragraph 3).

17. As per claim 7, Bebout ('404) discloses Debugging apparatus comprising:

a debugging processor (as shown in Figure 3, element 310);

a supplemental processor in communication with said debugging processor (as shown in Figure 3, element 340); and

a debugging program installed in said debugging processor and operable to,

activate a secondary program in said supplemental processor to transmit register states of said supplemental processor to said debugging processor (column 6 lines 57-60, interrogation of the processor is done using RPC commands) subsequent to the time a specified program, installed therein to be debugged, is operationally interrupted (column 6 lines 39-44, the processor is stopped before reading out its state),

modify parameters of said specified program (column 6 lines 57-60), and restore operation of said specified program in said supplemental processor with alterations as created in the debugging process (column 4 lines 66, the run command is available to resume operation).

Bebout ('404) discloses that the debugger can modify the state of its respective CPU (column 5 lines 10-14), that the debugger allows the user to diagnose the status of the system (column 6 lines 39-40) and that the debugger has an updateable state that can reflect the current state of a CPU (column 7 lines 13-15). However, Bebout ('404) does not expressly disclose the debugging apparatus wherein the debug processor has an available pool of memory that is used for debug operations.

Wikipedia's RAM teaches general information about random access memory.

At the time of invention it would have been obvious to a person of ordinary skill in the art to modify the debug system disclosed by Bebout ('404) such that the debugger stores CPU state information to a memory. This modification would have been obvious because computers use RAM to hold program code and data during execution (Wikipedia's RAM, first sentence under Overview).

Bebout ('404) in view of Wikipedia's RAM does not disclose the use of reserving memory.

Moreton ('630) teaches a method of arranging data within memory (see abstract).

At the time of invention it would have been obvious to a person of ordinary skill in the art to modify the debug system disclosed by Bebout ('404) in view of Wikipedia's RAM such that a set of addresses are reserved for debug operations. This modification would have been obvious because when a memory location is accessed its neighboring locations are accessed simultaneously (Moreton ('630) paragraph 3).

18. Claim 6 is rejected under 35 U.S.C. 103(a) as being unpatentable over Bebout ('404) in view of Wikipedia's RAM.

Bebout ('404) discloses Debugging apparatus comprising:

a debugging processor (as shown in Figure 3, element 310);

a supplemental processor (as shown in Figure 3, element 340);

a specified program to be debugged installed for operation in said supplemental processor (column 4 lines 9-13) and in an operationally interrupted state (column 6 lines 39-40); and

a communication path between and interconnecting said supplemental processor and said debugging processor (as shown in Figure 3, elements 322 and 323);

a secondary program in said supplemental processor operable, on command from said debugging processor, to transmit register states of said supplemental processor, at the time said specified program is operationally interrupted, to said debugging processor (column 6 lines 57-60, RPC commands are used to interrogate the state of the supplemental processor).

Bebout ('404) discloses that the debugger can modify the state of its respective CPU (column 5 lines 10-14), that the debugger allows the user to diagnose the status of the system (column 6 lines 39-40) and that the debugger has an updateable state that can reflect the current state of a CPU (column 7 lines 13-15). However, Bebout ('404) does not expressly disclose the debugging apparatus wherein the debug processor has an available pool of memory that is used for debug operations.

Wikipedia's RAM teaches general information about random access memory.

At the time of invention it would have been obvious to a person of ordinary skill in the art to modify the debug system disclosed by Bebout ('404) such that the debugger stores CPU state information to a memory. This modification would have been obvious because computers use RAM to hold program code and data during execution (Wikipedia's RAM, first sentence under Overview).

Conclusion

The prior art made of record on accompanying PTO 892 form and not relied upon is considered pertinent to applicant's disclosure. Specifically, Lee ('274) teaches a processor that repeatedly stops and saves its execution state with the possibility of instruction fix injection into its processing state and Austen ('470) teaches a system that stops all processors upon detecting a corruption and audits their registers to detect and correct a problem.

Contact Information

JS

[illegible]
SCOTT BADERMAN
SUPERVISORY PATENT EXAMINER